Laporan Tugas Kecil 2 IF2211:
Penerapan Algoritma *Decrease and Conquer* dalam *Topological Sorting* Pemilihan Mata Kuliah

Oleh:
Muhammad Tito Prakasa - K1 - 13519007

I. Keterkaitan *Topological Sort* dan Algoritma *DnC*

Decrease and conquer adalah algoritma yang mirip dengan *divide and conquer*. Hanya saja jika *divide and conquer* akan membagi persoalan menjadi dua sub-persoalan dan menyelesaikan keduanya kemudian digabungkan sehingga membentuk solusi dari persoalan utama. *Decrease and conquer* tidak membagi persoalan melainkan "menyederhanakan" persoalan dengan terus membuat sub-persoalan dari persoalan utama sehingga ketika sub-persoalan sudah cukup sederhana untuk diselesaikan, solusi yang didapatkan dapat mewakili solusi dari persoalan utamanya.

Topological sorting adalah algoritma pengurutan dari sebuah *directed acylic graph* (DAG) menjadi urutan linier masing-masing simpul graph dari mata kuliah yang harus diambil terlebih dahulu (tidak memiliki panah masuk sama sekali) hingga mata kuliah yang memiliki *prerequisites* sebelum bisa diambil (memiliki panah masuk yang banyak).

Pendekatan *topological sorting* sesuai dengan konsep *decrease and conquer*, yaitu menyederhanakan persoalan dan menyelesaikan sub-persoalannya. Hal ini dapat ditelusuri dari runutan algortima *topological sorting* seperti di bawah ini:

1. Cari simpul yang tidak memiliki panah masuk
2. Hapus simpul tersebut dan juga sisi yang berhubungan dengan simpul tersebut
3. Masukkan simpul tersebut ke dalam urutan linier dengan asumsi semakin kiri urutan maka semakin utama pula mata kuliah tersebut diambil terlebih dahulu..
4. Lakukan kembali langkah 1, 2, dan 3 hingga semua simpul habis.
5. Dari langkah-langkah tersebut maka akan dihasilkan urutan linier mata kuliah yang harus diambil terlebih dahulu.

II. *Source Code*

- Fungsi makeGraph(FILE raw):

```
#Fungsi untuk membuat graph dari raw file reader
def makeGraph(raw):
    graphDict = {}
    for value in raw:
        arrayOfCourse = value.strip('\n.').split(',')
        keyDict = arrayOfCourse[0]
        del arrayOfCourse[0]
        graphDict[keyDict] = arrayOfCourse
    return graphDict
```

- Fungsi deleteEdges(string simpulTerhapus, graph rawGraph):

```
#Fungsi untuk menghapus edge yang verticenya telah dihapus
def deleteEdges(verticeDeleted,rawGraph):
    counter = 0
    verticeWillDeleted = []

    for unit in rawGraph:
        if (verticeDeleted in rawGraph.get(unit)):
            indexToBeDeleted = rawGraph.get(unit).index(verticeDeleted)
            del rawGraph[unit][indexToBeDeleted]
```

- Fungsi findNull(graph rawGraph):

```
#Fungsi yang mencari vertice dengan edge kosong dan vertice tersebut akan dihapus. Begitu juga edge-edge yang terhubung dengan vertice
yang dihapus
def findNull(rawGraph):

    returnArray = []
    counter=0

    for unit in rawGraph:
        if (len(rawGraph.get(unit))==0):
            returnArray = returnArray + [unit]
            counter+=1

    #del vertice
    for i in range(counter):
        del rawGraph[returnArray[i]]

    #del edge
    for i in range(counter):
        deleteEdges(returnArray[i],rawGraph)

    #return nilai vertice yang kosong
    return returnArray
```

- Fungsi topoSort(array hasilSort, graph rawGraph, int limit):

```
#Membuat topo sort dari rawGraph dan hasilnya akan disimpan dalam array sorted. Limit adalah penanda
apakah topoSort dicukupkan atau tidak (8 semester maks).
def topoSort(sorted,rawGraph,limit):
    if (limit==7):
        sorted.append(findNull(rawGraph))
    else:
        if (len(rawGraph)!=0):
            sorted.append(findNull(rawGraph))
            topoSort(sorted,rawGraph,limit+1)
```

III. *Test Case*

1. General:

```
1   C1,C3.
2   C2,C1,C4.
3   C3.
4   C4,C1,C3.
5   C5,C2,C4.
```

```
Semester 1: C3
Semester 2: C1
Semester 3: C4
Semester 4: C2
Semester 5: C5


Tekan enter untuk menyudahi program.
```

2. Melebihi Batas (8 Semester):

```
1   C1,C3.
2   C2,C1,C4.
3   C3.
4   C4,C1,C3.
5   C5,C2,C4.
6   C6,C5.
7   C7,C6.
8   C8,C7.
9   C9,C7.
10  C10,C9.
```

```
Semester 1: C3
Semester 2: C1
Semester 3: C4
Semester 4: C2
Semester 5: C5
Semester 6: C6
Semester 7: C7
Semester 8: C8, C9


Tekan enter untuk menyudahi program.
```

3. Mata Kuliah ITB:

```
1   MA1110.
2   MA1210.
3   FI1110.
4   FI1210.
5   KI1110.
6   IF2124,MA1110,MA1210,FI1110.
7   IF2121,MA1110,MA1210,FI1110.
8   IF2123,MA1110,MA1210,FI1110.
9   IF2130,MA1110,MA1210,FI1110.
10  IF2110,MA1110,MA1210,FI1110.
11  IF2120,MA1110,MA1210,FI1110.
12  IF2211,IF2121.
13  IF2250,IF2110.
14  IF2220,IF2120.
15  IF2230,IF2130.
16  IF2240,IF2120.
17  IF2210,IF2110.
```

```
Semester 1: MA1110, MA1210, FI1110, FI1210, KI1110
Semester 2: IF2124, IF2121, IF2123, IF2130, IF2110, IF2120
Semester 3: IF2211, IF2250, IF2220, IF2230, IF2240, IF2210


Tekan enter untuk menyudahi program.
```

4. 1 Simpul:

```
1   C2.
```

```
Semester 1: C2

Tekan enter untuk menyudahi program.
```

5. Kosong:

```
1
```

```
Tekan enter untuk menyudahi program.
```

6. Banyak Simpul Tidak Ada Sisi:

```
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
```

```
Semester 1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12


Tekan enter untuk menyudahi program.
```

7. Terdapat Directed Cycle:

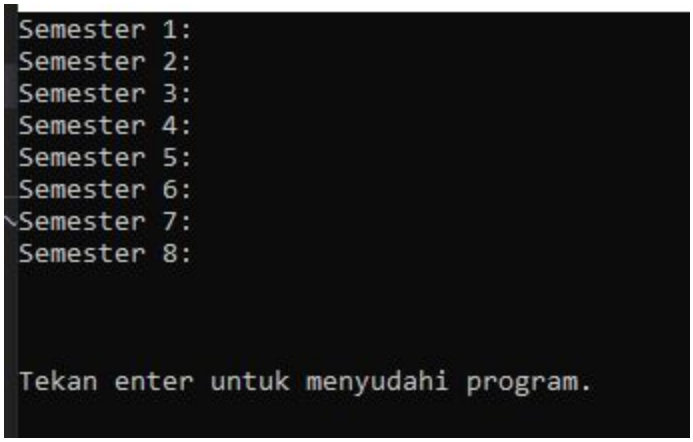
```
1.
2,1.
3,2,1.
4,3,2,1.
5,4,3,2,1.
6,5,4,3,2,1.
7,6,5,4,3,2,1.
8,7,6,5,4,3,2,9.
9,8,7,6,5,4,3,2,1.
```

```
Semester 1: 1
Semester 2: 2
Semester 3: 3
Semester 4: 4
Semester 5: 5
Semester 6: 6
Semester 7: 7
Semester 8:


Tekan enter untuk menyudahi program.
```

8. 1 Simpul Bersisian dengan Simpul yang Tidak Ada:

```
1,2,3,4,5.
```



IV. Alamat *Source Code*

grevicoc/topological-sort-implementation (github.com)

V. Kesimpulan

Poin	Ya	Tidak
1. Program Berhasil dikompilasi	✔	
2. Program berhasil dirunning	✔	
3. Program dapat menerima berkas input dan menuliskan output	✔	
4. Luaran sudah benar untuk semua input	✔	